THRIVER ASHISH'S

DATA STRUCTURES AND ALGORITHM SHEET

This 90 days Roadmap covers all the topics of DS-Algo which are most important for technical interviews and also important to become a good software engineer

This sheet also covers all the questions that are needed to be solved in order to crack tech giants interviews

90 DAYS

https://www.youtube.com/c/thriverashish

https://www.youtube.com/c/thrivewithashish

https://www.linkedin.com/in/thriverashish

@thriverashish

THRIVE WITH ASHISH

ARRAYS

@thriverashish

10 DAYS

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

DAY 1

Understand Big O notation (Time and Space complexity)

DAY 2

-- Study Basic Concepts of Array and get familiar with List DS in the programming language you chose.

-- Reverse the array in place (space complexity should be constant)
Input —>> 3,5,9,4,2
Output —>> 2,4,9,5,3

-- Insert an element in between of array

DAY 3 AND DAY 4

- Understand Quick Sort, Merge Sort, Insertion Sort and Selection Sort
- Implement them all in your fav programming language

DAY 5

- Understand Binary Search Algorithm
- Search an element in the Sorted Array
- Leetcode Question 33 --> Search in Rotated Sorted Array
- Leetcode Question 50 --> Implement pow(x, n), which calculates x raised to the power n (i.e., xn)

DAY 6

- Understand 2D matrix/ @D Arrays
- Search a 2D Matrix Leetcode 74
- Set Matrix to Zero —> Leetcode 73
- Pascals Triangle —> Leetcode 118
- Rotate Matrix —> Leetcode 48

DAY 7

- Kadane's Algo —> Maximum Subbarray Leetcode 73
- Sort Array of 0's 1's and 2's Leetcode 75
- Two Sum Problem —> Leetcode 1
- Find Duplicate

THRIVE WITH ASHISH

@thriverashish

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

DAY 8

- Stock Buy and Sell --> Leetcode 121
- Next Permutation --> Leetcode 32
- Merge Intervals --> Leetcode 56
- Merge Sorted Arrays --> Leetcode 88

DAY 9

- Reverse pairs --> Leetcode 493
- Grid Unique Paths --> Leetcode 62
- 3 Sum problem --> Leetcode 15
- 4 Sum Problem --> Leetcode 18

DAY 10

- Repeat Missing Number Array --> InterviewBit
  https://www.interviewbit.com/problems/repeat-and-missing-number-array/
- Longest Consecutive Subsequence —>Leetcode 128
- Subarray with given Xor —> Interview Bit
  https://www.interviewbit.com/problems/subarray-with-given-xor/
- Longest Substring without repeat —>

THRIVE WITH ASHISH

LINKED LIST

@thriverashish

04 DAYS

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

DAY 11

- Understand the concept of Linked List
- Understand its Advantages and disadvantages over Arrays
- Implement the Linked list in the language you chose:
- Create Linked List class
  Create a Node
  Add Node
  Remove Node
  Find Node in Linked List

DAY 12

- Reverse Linked List
- Find Middle of the Linked List
- Remove Nth Node From End of List
- Merge Two Sorted Lists

DAY 13

- Add Two Numbers given in linked list
- Detect and remove Linked List Cycle
- Intersection of Two Linked Lists
- Palindrome Linked List

DAY 14

- Reverse Nodes in k-Group
- Rotate List
- Flattening a Linked List
- Copy List with Random Pointer

THRIVE WITH ASHISH

STACK

@thriverashish

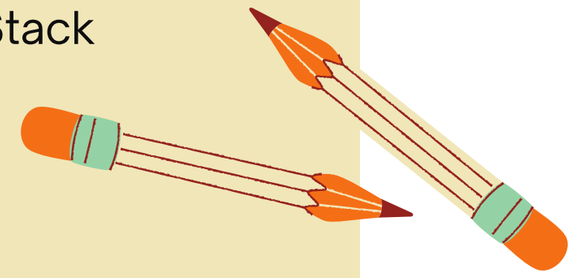
https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

03 DAYS

DAY 15

- Understand the basics of Stack Data Structures
- Relate with some real-life examples like
- Stack of Note books and Stack of coins
- Implement the Stack using Arrays (Push Pop etc functionalities)
- Implement the Stack with Linked List



DAY 16

- Sort a Stack
- Valid Parenthesis --> Leetcode 20
- Retrieve Minimum Element from the Stack --> Leetcode 155

DAY 17 (LEVEL UP)

- Next Greater Element using Stack --> Leetcode 496
- Find largest rectangle in Histogram --> Leetcode 84

THRIVE WITH ASHISH

QUEUES

@thriverashish

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

03 DAYS

DAY 18

- Understand the basics of Queue Data Structures
- Relate with some real-life examples
- Implement the Queue using Arrays
- Implement the Queue using the Linked List

DAY 19

- Implement Queue using Stack
- Implement Stack using Queue
- Implement and understand Double Ended Queue

DAY 20

- Understand and Implement Circular Queue
- Fine examples where circular Queues can be used.
- Implement Producer Consumer Problem

DAY 21

- Queue Reversal
- Implement LRU Cache
- Implement LFU Cache

DAY 22

- Learn Sliding window Coding Pattern
- Sliding Window Maximum Leetcode 239
- First negative integer in every window of size

DAY 23

- Trapping Rain Water (Stack-based) Leetcode 42
- Simplify Directory Path (Stack Based) Leetcode 71
(No Priority Queue for now)

THRIVE WITH ASHISH

RECURSION

@thriverashish

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

04 DAYS

DAY 24

- Understand Recursion and its base conditions etc.
- Understand in depth not only what recursion is but also what kind of Data str it uses internally etc.
- You will get to know how StackOverflow error occurs

DAY 25

- Fibonacci Number
- Reverse the Linked List using the Recursive approach
- Reverse LinkedList in k-groups

DAY 26

- Subset Sums
- Subset Sums II

DAY 27

- Combination Sum
- Combination Sum II
- Palindrome Partitioning
- Permutation Sequence

THRIVE WITH ASHISH

REVISION

@thriverashish

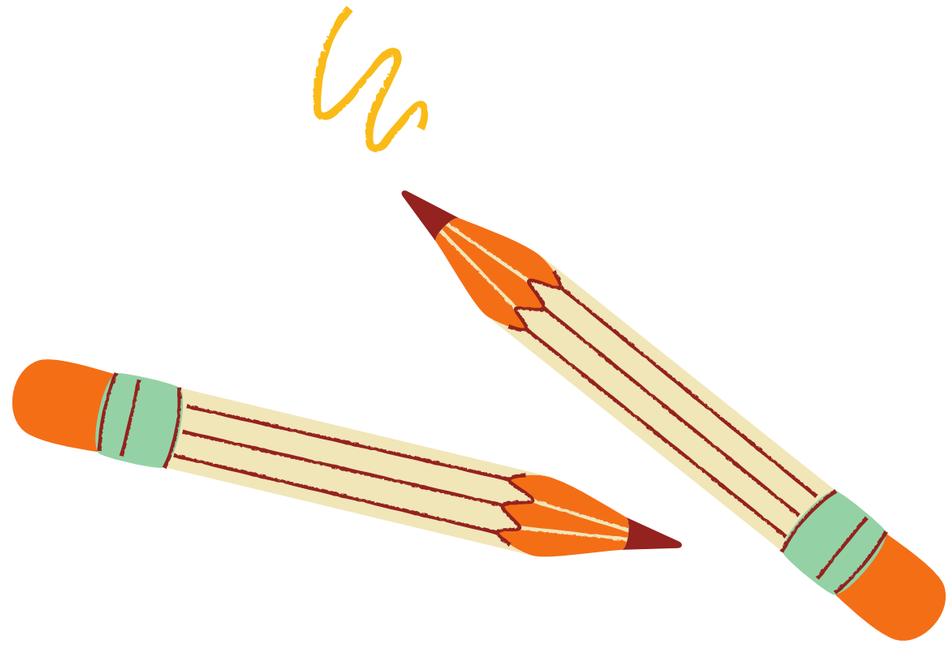
https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

03 DAYS

DAY 28

Revise Arrays and Linked List



DAY 29

Revise Queues and Stacks

DAY 30

Revise Recursion and practice

THRIVE WITH ASHISH

HASHING AND HASH BASED DS

@thriverashish
https://www.linkedin.com/in/thriverashish
https://www.youtube.com/c/thriverashish

04 DAYS

DAY 31

- Understand what exactly Hash-based Data structures are!
- What is hashing?
- What is a hash code?
- What is Hash Function?
- Understand the concept of ROLLING HASH
- Understand the usage and application of Hash Tables like
  -- Used to implement database indexes.
  -- Used to implement associative arrays.
  -- Used to implement the different data structures like hashMap/HashSet etc

DAY 32

- Understand the internals of HashMap and HashSet
- Read the code of the already implemented hashMap/HashSet library if the code is available like in Java.
- Do a bit of hands-on, iterate the hashMap/set etc

DAY 33

- Design HashSet -- Leetcode 705
- Design HashMap -- Leetcode 706
- Two Sum -- Leetcode 1
- Running Sum of 1D Array Leetcode 1480

DAY 34

- Check If a String Contains All Binary Codes of Size K -- Leetcode 1461
- K Divisible Elements Subarrays -- Leetcode 2261
- Number of Distinct Islands -- Leetcode 711

THRIVE WITH ASHISH

TREES AND TRIES

@thriverashish

10 DAYS

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

DAY 35

- Read the theory of Binary Trees and different types of Binary Trees
- Understand the Node structure of Binary Tree

  Understand and Write Iterative code for following Traversals

- Level Order Traversal
- Inorder Traversal - Leetcode 194
- PreOrder Traversal
- Postorder Traversal
- Zig Zag Level Oder Traversal

DAY 36

- Implement Binary Tree
- Add Node
- Remove Node
- Traverse Node
- Construct Binary Tree from Inorder and PostOrder - Leetcode 105

DAY 37

- Same Tree - Leetcode 100
- Invert Binary Tree - Leetcode 226
- Symmetric Binary Tree - Leetcode 101
- Left/Bottom/Top View of Binary Tree
- Vertical Order Traversal of a Binary Tree - Leetcode 987

DAY 38

- Maximum Width of Binary Tree - Leetcode 622
- Maximum Depth of Binary Tree - Leetcode 104
- Diameter of Binary Tree - Leetcode 543
- Balanced Binary Tree - Leetcode 110
- Lowest Common Ancestor of a Binary Tree - Leetcode 236

DAY 39

- Identify the Path to the given Node
- Binary Tree Maximum Path Sum - Leetcode 124
- Flatten Binary Tree to Linked List - Leetcode 1114
- Mirror Binary Tree
- Serialize and Deserialize Binary Tree - Leetcode 297
- Vertical Sum of Nodes in Binary Tree

THRIVE WITH ASHISH

@thriverashish

10 DAYS

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

DAY 40

- Understand Binary Search Tree
- Search an Element in BST - Leetcode 700
- Understand the Complexity of searching in BST
- Find Lowest Common Ancestor of a given node in BST - Leetcode 235
- Convert Sorted Array to Binary Search Tree - Leetcode 108
- Validate Binary Search Tree - Leetcode 98

DAY 41

- Construct BST from Preorder Traversal - Leetcode 1008
- Recover BST - Leetcode 99
- Identify Predecessor and Successor of a Node in BST
- Kth Smallest Element in a BST
- Two Sum - Input is BST - Leetcode 653

DAY 42

- Understand Self-Balancing Binary Search Trees
- Read and Understand Red Black Tree
- Read and Understand AVL Tree

DAY 43

- Understand about Quad trees and N-Ary Trees
- Understand Trie Data Structure
- Implement Trie (Prefix Tree)

DAY 44

- Maximum XOR of Two Numbers in an Array - Leetcode 421
- Maximum XOR With an Element From Array - Leetcode 1707
- Hotel Reviews - InterviewBit

THRIVE WITH ASHISH

HEAP

@thriverashish

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

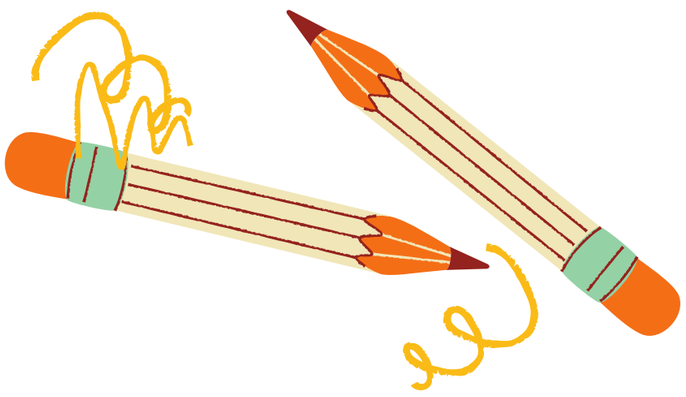
03 DAYS

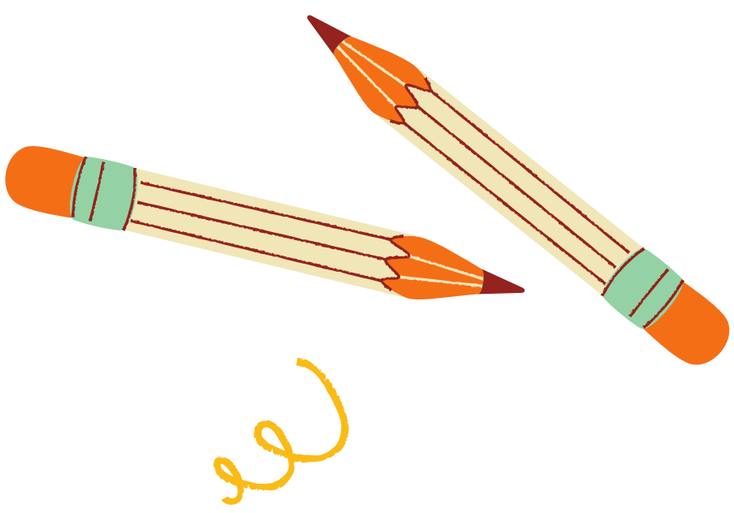
DAY 45

- Understand what HEAP DS is all about like properties and all
  (heap is a complete Binary Tree)

- Implement its Operations (Min and Max Heap)
  - Heapify
  - Insert Element into Heap
  - Delete Element from Heap
  - Peek
  - Extract Min/Max

- Understand and Read Heap Data Structure Applications
  -Heap is used while implementing a priority queue.
  -Dijkstra's Algorithm
  -Heap Sort



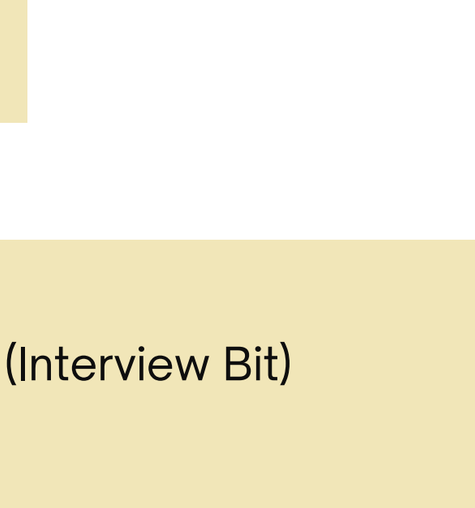


DAY 46

- Distinct Numbers in the window (Interview Bit)
- Merge K Sorted Arrays
- Kth Largest Element in an Array - Leetcode 215

DAY 47

- Top K Frequent Elements - Leetcode 347
- Maximum Sum Combinations - InterviewBit
- Find Median from Data Stream (Hard) - Leetcode 295



THRIVE WITH ASHISH

GRAPH

@thriverashish

10 DAYS

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

DAY 48

- Understand Graphs
- Understand all three Graph representations and do hands-on
  - Edge list
  - Adjacency Matrix
  - Adjacency List
- Read and Understand Different types of Graphs
  - Directed Graphs
  - Weighted Graphs etc

DAY 49

- Breadth-first search (BFS)
- Depth-first search (DFS)

DAY 50

- Dijkstra - The shorter path from a given node to all vertices
- Floyd Warshall - The shorter path from every vertex to every other vertex

DAY 51

- Prims - Minimum spanning tree
- Kruskal - Minimum spanning tree

DAY 52

- Topological sort
- Articulation points in a graph
- Bridges in a graph
- Johnsons algorithm

DAY 53

- Clone Graph - Leetcode 133
- Number of Islands -- Leetcode 200
- Course Schedule -- Leetcode 207

DAY 54

- Valid path -- Leetcode 1391
- Is Graph Bipartite? -- Leetcode 785
- Smallest multiple with 0 & 1

DAY 55

- Commutable islands
- Detect cycle in Directed and Undirected Graph
- Black shapes

DAY 56

- Knight on chess board
- Word ladder I
- Word ladder II

DAY 57

- Smallest sequence with given primes
- Capture regions on board
- Word search board

THRIVE WITH ASHISH

REVISION

@thriverashish

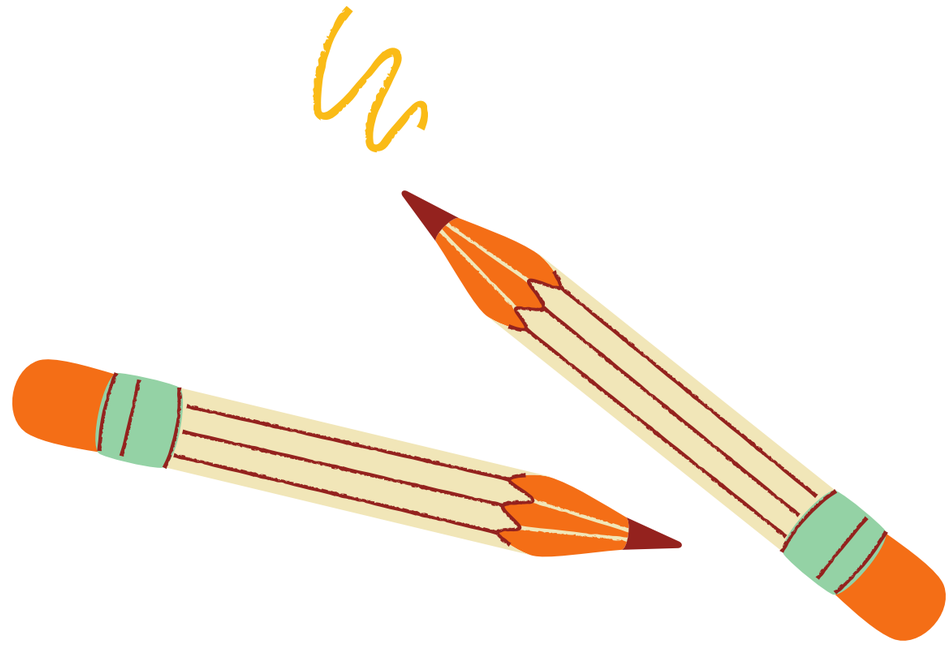
https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

03 DAYS

DAY 58

Revise Hashing and Practice



DAY 59

Revise Trees and Tries

DAY 60

Revise Graphs and practice

THRIVE WITH ASHISH

GREEDY

@thriverashish

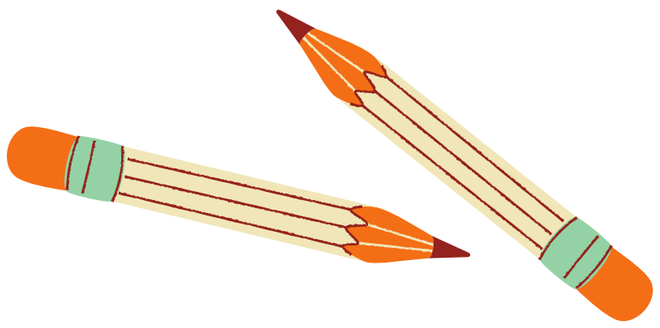
https://www.linkedin.com/in/thriverashish

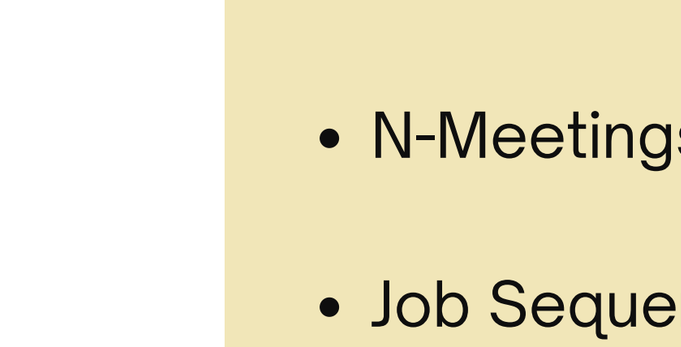
https://www.youtube.com/c/thriverashish

02 DAYS

DAY 61

- Fractional Knapsack problem
- N-Meetings in one room
- Job Sequencing Problem





DAY 62

- Minimum Platforms
- Find Minimum Number

THRIVE WITH ASHISH

DYNAMIC PROGRAMMING

10 DAYS

@thriverashish

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

DAY 63

- Understand what overlapping Subproblems mean?
- Understand memoization
- Fibonacci Number
- Catalan Number

DAY 64

- Maximum Product Subarray Leetcode 152
- Longest Increasing Subsequence Leetcode 300
- Longest Common Subsequence Leetcode 1143

DAY 65

- Ones and Zeroes Leetcode 474
- Edit Distance Leetcode 72
- Minimum Path Sum Leetcode 64

DAY 66

- Coin Change Leetcode 322
- Partition Equal Subset Sum Leetcode 416
- Minimum Cost to Cut a Stick Leetcode 1547

DAY 67

- Ways to decode
- Stairs
- Jump game

DAY 68

- Min jumps
- Longest AP
- Evaluate expression to true

DAY 69

- Best time to buy and sell stocks - 1
- Min sum path in matrix

DAY 70

- Max rectangle in binary matrix
- Rod cutting
- Coin sum infinite

DAY 71

- Best time to buy and sell stocks II
- Arrange II
- Best time to buy and sell stocks III

DAY 72

- Equal avg partition
- Regex match
- Regex match II

DAY 73

- Unique BST II
- Max sum path in BT
- Palindromic partitioning

THRIVE WITH ASHISH

BACKTRACKING

@thriverashish

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

05 DAYS

DAY 74

- Understand why Backtracking?
  - when should we apply backtracking
  - how should we apply backtracking

Solve Following Questions

- Subset I
- Subset II

DAY 75

- Combination sum -- Leetcode 39
- Combination sum II -- Leetcode 40

DAY 76

- Palindrome Partitioning -- Leetcode 131
- Generate all parenthesis
- NQueens -- Leetcode 51

DAY 77

- Sudoku Solver -- Leetcode 37
- Permutations -- Leetcode 46
- Kth permutation sequence Leetcode -- 60

DAY 78

- Gray code
- M-Coloring Problem
- Rat in a Maze

THRIVE WITH ASHISH

STRINGS

@thriverashish

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

04 DAYS

DAY 79

- Reverse Words in a String
  Leetcode 151
- Longest Palindromic Substring
  Leetcode 5
- Roman to Integer Leetcode 13

DAY 80

- String to Integer (atoi)
  Leetcode 8
- Longest Common Prefix
  Leetcode 14
- Rabin Carp

DAY 81

- Implement strStr() {Z Function}
  Leetcode 28
- KMP algo / LPS(pi) array -- Leetcode 28
- Minimum Characters required to make a String Palindromic -- Inerview bit

DAY 82

- Valid Anagram -- Leetcode 242
- Count and Say -- Leetcode 38
- Compare Version Numbers -- Leetcode 165

THRIVE WITH ASHISH

RANDOM QUESTIONS

@thriverashish

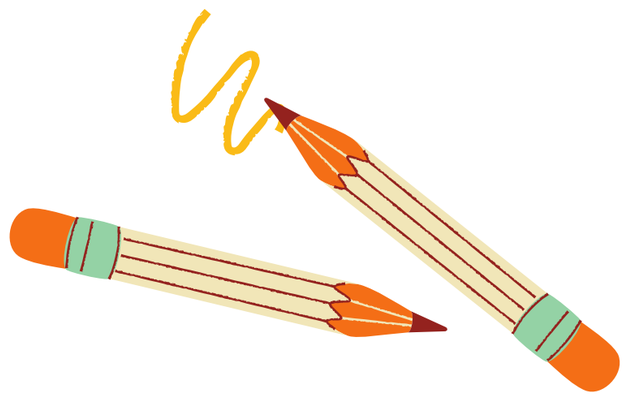
08 DAYS

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

DAY 83-85

Solve Random Array, Linked List, Stacks and Queue Questions



DAY 86-87

Solve Random Recursion Hashing, Tree, and Graph Questions

DAY 88-90

Solve Random DP, Backtracking, Strings, and other Questions

DAY ...

Start Applying, you are Ready

THRIVE WITH ASHISH

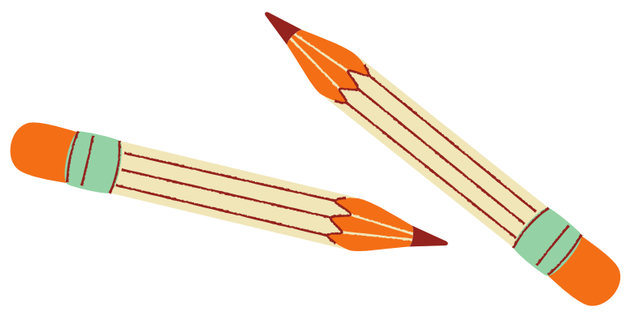
GOOD LUCK

@thriverashish

https://www.linkedin.com/in/thriverashish

https://www.youtube.com/c/thriverashish

All the best!



THRIVE WITH ASHISH

Oracle SQL Cheat Sheet

www.databasestar.com

SELECT Query

```
SELECT col1, col2
FROM table
JOIN table2 ON table1.col = table2.col
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY col1 ASC|DESC;
```

SELECT Keywords

DISTINCT: Removes duplicate results

```
SELECT DISTINCT product_name
FROM product;
```

BETWEEN: Matches a value between two other values (inclusive)

```
SELECT product_name
FROM product
WHERE price BETWEEN 50 AND 100;
```

IN: Matches to any of the values in a list

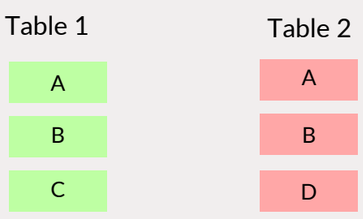
```
SELECT product_name
FROM product
WHERE category IN
('Electronics', 'Furniture');
```

LIKE: Performs wildcard matches using _ or %

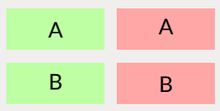
```
SELECT product_name
FROM product
WHERE product_name
LIKE '%Desk%";
```

Joins

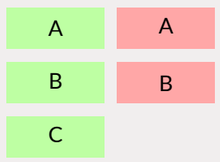
```
SELECT t1.*, t2.*
FROM t1
join_type t2 ON t1.col = t2.col;
```



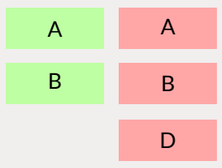
INNER JOIN: show all matching records in both tables.



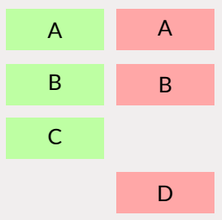
LEFT JOIN: show all records from left table, and any matching records from right table.



RIGHT JOIN: show all records from right table, and any matching records from left table.



FULL JOIN: show all records from both tables, whether there is a match or not.



CASE Statement

Simple Case

```
CASE name
  WHEN 'John' THEN 'Name John'
  WHEN 'Steve' THEN 'Name Steve'
  ELSE 'Unknown'
END
```

Searched Case

```
CASE
  WHEN name='John' THEN 'Name John'
  WHEN name='Steve' THEN 'Name Steve'
  ELSE 'Unknown'
END
```

Common Table Expression

```
WITH queryname AS (
  SELECT col1, col2
  FROM firsttable)
SELECT col1, col2..
FROM queryname...;
```

Modifying Data

Insert

```
INSERT INTO tablename
(col1, col2...)
VALUES (val1, val2);
```

Insert from a Table

```
INSERT INTO tablename
(col1, col2...)
SELECT col1, col2...
```

Insert Multiple Rows

```
INSERT
INTO tablename (col1, col2)
VALUES (valA1, valB1)
INTO tablename (col1, col2)
VALUES (valA2, valB2)
SELECT * FROM dual;
```

Update

```
UPDATE tablename
SET col1 = val1
WHERE condition;
```

Update with a Join

```
UPDATE t
SET col1 = val1
FROM tablename t
INNER JOIN table x
ON t.id = x.tid
WHERE condition;
```

Delete

```
DELETE FROM tablename
WHERE condition;
```

Indexes

Create Index

```
CREATE INDEX indexname
ON tablename (cols);
```

Drop Index

```
DROP INDEX indexname;
```

Set Operators

UNION: Shows unique rows from two result sets.

UNION ALL: Shows all rows from two result sets.

INTERSECT: Shows rows that exist in both result sets.

EXCEPT: Shows rows that exist in the first result set but not the second.

Aggregate Functions

- SUM: Finds a total of the numbers provided
- COUNT: Finds the number of records
- AVG: Finds the average of the numbers provided
- MIN: Finds the lowest of the numbers provided
- MAX: Finds the highest of the numbers provided

Common Functions

- LENGTH(string): Returns the length of the provided string
- INSTR(string, substring, [start_position], [occurrence]): Returns the position of the substring within the specified string.
- TO_CHAR(input_value, [fmt_mask], [nls_param]): Converts a date or a number to a string
- TO_DATE(charvalue, [fmt_mask], [nls_date_lang]): Converts a string to a date value.
- TO_NUMBER(input_value, [fmt_mask], [nls_param]): Converts a string value to a number.
- ADD_MONTHS(input_date, num_months): Adds a number of months to a specified date.
- SYSDATE: Returns the current date, including time.
- CEIL(input_val): Returns the smallest integer greater than the provided number.
- FLOOR(input_val): Returns the largest integer less than the provided number.
- ROUND(input_val, round_to): Rounds a number to a specified number of decimal places.
- TRUNC(input_value, dec_or_fmt): Truncates a number or date to a number of decimals or format.
- REPLACE(whole_string, string_to_replace, [replacement_string]): Replaces one string inside the whole string with another string.
- SUBSTR(string, start_position, [length]): Returns part of a value, based on a position and length.

Create Table

Create Table

```
CREATE TABLE tablename (
  column_name data_type
);
```

Create Table with Constraints

```
CREATE TABLE tablename (
  column_name data_type NOT NULL,
  CONSTRAINT pkname PRIMARY KEY (col),
  CONSTRAINT fkname FOREIGN KEY (col)
REFERENCES other_table(col_in_other_table),
  CONSTRAINT ucname UNIQUE (col),
  CONSTRAINT ckname CHECK (conditions)
);
```

Create Temporary Table

```
CREATE GLOBAL TEMPORARY TABLE
tablename (
  colname datatype
) ON COMMIT DELETE ROWS;
```

Drop Table

```
DROP TABLE tablename;
```

Alter Table

Add Column

```
ALTER TABLE tablename
ADD columnname datatype;
```

Drop Column

```
ALTER TABLE tablename
DROP COLUMN columnname;
```

Modify Column

```
ALTER TABLE tablename MODIFY
columnname newdatatype;
```

Rename Column

```
ALTER TABLE tablename RENAME COLUMN
currentname TO newname;
```

Add Constraint

```
ALTER TABLE tablename ADD
CONSTRAINT constraintname
constrainttype (columns);
```

Drop Constraint

```
ALTER TABLE tablename DROP
constraint_type constraintname;
```

Rename Table

```
sp_rename
'old_table_name',
'new_table_name';
```

Window/Analytic Functions

```
function_name ( arguments ) OVER (
[query_partition_clause]
[ORDER BY order_by_clause
[windowing_clause] ] )
```

Example using RANK, showing the student details and their rank according to the fees_paid, grouped by gender:

```
SELECT
student_id, first_name, last_name, gender, fees_paid,
RANK() OVER (
  PARTITION BY gender ORDER BY fees_paid
) AS rank_val
FROM student;
```

Subqueries

Single Row

```
SELECT id, last_name, salary
FROM employee
WHERE salary = (
  SELECT MAX(salary)
  FROM employee
);
```

Multi Row

```
SELECT id, last_name, salary
FROM employee
WHERE salary IN (
  SELECT salary
  FROM employee
  WHERE last_name LIKE 'C%'
);
```

SQL Server Cheat Sheet

www.databasestar.com

SELECT Query

```
SELECT col1, col2
FROM table
JOIN table2 ON table1.col = table2.col
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY col1 ASC|DESC;
```

SELECT Keywords

DISTINCT: Removes duplicate results

```
SELECT DISTINCT product_name
FROM product;
```

BETWEEN: Matches a value between two other values (inclusive)

```
SELECT product_name
FROM product
WHERE price BETWEEN 50 AND 100;
```

IN: Matches to any of the values in a list

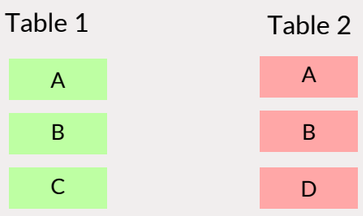
```
SELECT product_name
FROM product
WHERE category IN
('Electronics', 'Furniture');
```

LIKE: Performs wildcard matches using _ or %

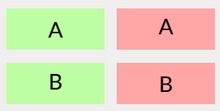
```
SELECT product_name
FROM product
WHERE product_name
LIKE '%Desk%";
```

Joins

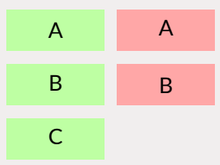
```
SELECT t1.*, t2.*
FROM t1
join_type t2 ON t1.col = t2.col;
```



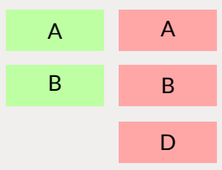
INNER JOIN: show all matching records in both tables.



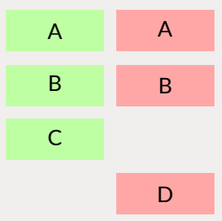
LEFT JOIN: show all records from left table, and any matching records from right table.



RIGHT JOIN: show all records from right table, and any matching records from left table.



FULL JOIN: show all records from both tables, whether there is a match or not.



CASE Statement

Simple Case

```
CASE name
  WHEN 'John' THEN 'Name John'
  WHEN 'Steve' THEN 'Name Steve'
  ELSE 'Unknown'
END
```

Searched Case

```
CASE
  WHEN name='John' THEN 'Name John'
  WHEN name='Steve' THEN 'Name Steve'
  ELSE 'Unknown'
END
```

Common Table Expression

```
WITH queryname (col1, col2...) AS (
  SELECT col1, col2
  FROM firsttable)
SELECT col1, col2..
FROM queryname...;
```

Modifying Data

Insert

```
INSERT INTO tablename
(col1, col2...)
VALUES (val1, val2);
```

Insert from a Table

```
INSERT INTO tablename
(col1, col2...)
SELECT col1, col2...
```

Insert Multiple Rows

```
INSERT INTO tablename
(col1, col2...) VALUES
(valA1, valB1),
(valA2, valB2),
(valA3, valB3);
```

Update

```
UPDATE tablename
SET col1 = val1
WHERE condition;
```

Update with a Join

```
UPDATE t
SET col1 = val1
FROM tablename t
INNER JOIN table x
ON t.id = x.tid
WHERE condition;
```

Delete

```
DELETE FROM tablename
WHERE condition;
```

Indexes

Create Index

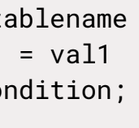
```
CREATE INDEX indexname
ON tablename (cols);
```

Drop Index

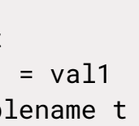
```
DROP INDEX indexname;
```

Set Operators

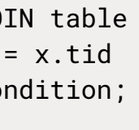
UNION: Shows unique rows from two result sets.



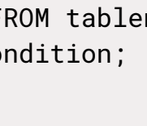
UNION ALL: Shows all rows from two result sets.



INTERSECT: Shows rows that exist in both result sets.



MINUS: Shows rows that exist in the first result set but not the second.



Aggregate Functions

- SUM: Finds a total of the numbers provided
- COUNT: Finds the number of records
- AVG: Finds the average of the numbers provided
- MIN: Finds the lowest of the numbers provided
- MAX: Finds the highest of the numbers provided

Common Functions

- LEN(string): Returns the length of the provided string
- CHARINDEX(string, substring, [start_position], [occurrence]): Returns the position of the substring within the specified string.
- CAST(expression AS type [(length)]): Converts an expression to another data type.
- GETDATE: Returns the current date, including time.
- CEILING(input_val): Returns the smallest integer greater than the provided number.
- FLOOR(input_val): Returns the largest integer less than the provided number.
- ROUND(input_val, round_to, operation): Rounds a number to a specified number of decimal places.
- REPLACE(whole_string, string_to_replace, replacement_string): Replaces one string inside the whole string with another string.
- SUBSTRING(string, start_position, [length]): Returns part of a value, based on a position and length.

Create Table

Create Table

```
CREATE TABLE tablename (
  column_name data_type
);
```

Create Table with Constraints

```
CREATE TABLE tablename (
  column_name data_type NOT NULL,
  CONSTRAINT pkname PRIMARY KEY (col),
  CONSTRAINT fkname FOREIGN KEY (col)
REFERENCES other_table(col_in_other_table),
  CONSTRAINT ucname UNIQUE (col),
  CONSTRAINT ckname CHECK (conditions)
);
```

Create Temporary Table

```
SELECT cols
INTO #tablename
FROM table;
```

Drop Table

```
DROP TABLE tablename;
```

Alter Table

Add Column

```
ALTER TABLE tablename
ADD columnname datatype;
```

Drop Column

```
ALTER TABLE tablename
DROP COLUMN columnname;
```

Modify Column

```
ALTER TABLE tablename ALTER COLUMN
columnname newdatatype;
```

Rename Column

```
sp_rename
'table_name.old_column_name',
'new_column_name', 'COLUMN';
```

Add Constraint

```
ALTER TABLE tablename ADD
CONSTRAINT constraintname
constrainttype (columns);
```

Drop Constraint

```
ALTER TABLE tablename
DROP CONSTRAINT constraintname;
```

Rename Table

```
ALTER TABLE tablename
RENAME TO newtablename;
```

Window/Analytic Functions

```
function_name ( arguments ) OVER (
[query_partition_clause]
[ORDER BY order_by_clause
[windowing_clause] ] )
```

Example using RANK, showing the student details and their rank according to the fees_paid, grouped by gender:

```
SELECT
student_id, first_name, last_name, gender, fees_paid,
RANK() OVER (
  PARTITION BY gender ORDER BY fees_paid
) AS rank_val
FROM student;
```

Subqueries

Single Row

```
SELECT id, last_name, salary
FROM employee
WHERE salary = (
  SELECT MAX(salary)
  FROM employee
);
```

Multi Row

```
SELECT id, last_name, salary
FROM employee
WHERE salary IN (
  SELECT salary
  FROM employee
  WHERE last_name LIKE 'C%'
);
```

MySQL Cheat Sheet

www.databasestar.com

SELECT Query

```
SELECT col1, col2
FROM table
JOIN table2 ON table1.col = table2.col
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY col1 ASC|DESC;
```

SELECT Keywords

DISTINCT: Removes duplicate results

```
SELECT DISTINCT product_name
FROM product;
```

BETWEEN: Matches a value between two other values (inclusive)

```
SELECT product_name
FROM product
WHERE price BETWEEN 50 AND 100;
```

IN: Matches to any of the values in a list

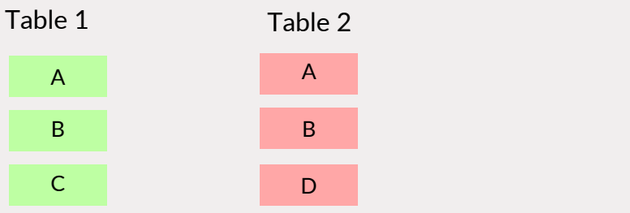
```
SELECT product_name
FROM product
WHERE category IN
('Electronics', 'Furniture');
```

LIKE: Performs wildcard matches using _ or %

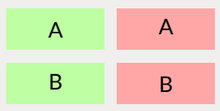
```
SELECT product_name
FROM product
WHERE product_name
LIKE '%Desk%";
```

Joins

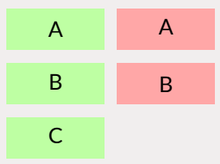
```
SELECT t1.*, t2.*
FROM t1
join_type t2 ON t1.col = t2.col;
```



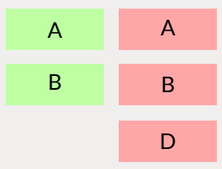
INNER JOIN: show all matching records in both tables.



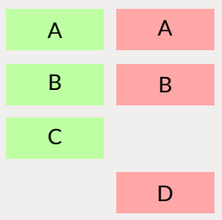
LEFT JOIN: show all records from left table, and any matching records from right table.



RIGHT JOIN: show all records from right table, and any matching records from left table.



FULL JOIN: show all records from both tables, whether there is a match or not.



CASE Statement

Simple Case

```
CASE name
  WHEN 'John' THEN 'Name John'
  WHEN 'Steve' THEN 'Name Steve'
  ELSE 'Unknown'
END
```

Searched Case

```
CASE
  WHEN name='John' THEN 'Name John'
  WHEN name='Steve' THEN 'Name Steve'
  ELSE 'Unknown'
END
```

Common Table Expression

```
WITH queryname AS (
  SELECT col1, col2
  FROM firsttable)
SELECT col1, col2..
FROM queryname...;
```

Modifying Data

Insert

```
INSERT INTO tablename
(col1, col2...)
VALUES (val1, val2);
```

Insert from a Table

```
INSERT INTO tablename
(col1, col2...)
SELECT col1, col2...
```

Insert Multiple Rows

```
INSERT INTO tablename (col1,
col2…)
VALUES
(valA1, valB1),
(valA2, valB2),
(valA3, valB3);
```

Update

```
UPDATE tablename
SET col1 = val1
WHERE condition;
```

Update with a Join

```
UPDATE t
SET col1 = val1
FROM tablename t
INNER JOIN table x
ON t.id = x.tid
WHERE condition;
```

Delete

```
DELETE FROM tablename
WHERE condition;
```

Indexes

Create Index

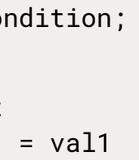
```
CREATE INDEX indexname
ON tablename (cols);
```

Drop Index

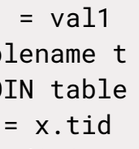
```
DROP INDEX indexname;
```

Set Operators

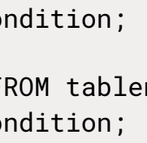
UNION: Shows unique rows from two result sets.



UNION ALL: Shows all rows from two result sets.



INTERSECT: Shows rows that exist in both result sets.



MINUS: Shows rows that exist in the first result set but not the second.

Aggregate Functions

- SUM: Finds a total of the numbers provided
- COUNT: Finds the number of records
- AVG: Finds the average of the numbers provided
- MIN: Finds the lowest of the numbers provided
- MAX: Finds the highest of the numbers provided

Common Functions

- LENGTH(string): Returns the length of the provided string
- INSTR(string, substring): Returns the position of the substring within the specified string.
- CAST(expression AS datatype): Converts an expression into the specified data type.
- ADDDATE(input_date, days): Adds a number of days to a specified date.
- NOW: Returns the current date, including time.
- CEILING(input_val): Returns the smallest integer greater than the provided number.
- FLOOR(input_val): Returns the largest integer less than the provided number.
- ROUND(input_val, [round_to]): Rounds a number to a specified number of decimal places.
- TRUNCATE(input_value, num_decimals): Truncates a number to a number of decimals.
- REPLACE(whole_string, string_to_replace, replacement_string): Replaces one string inside the whole string with another string.
- SUBSTRING(string, start_position): Returns part of a value, based on a position and length.

Create Table

Create Table

```
CREATE TABLE tablename (
  column_name data_type
);
```

Create Table with Constraints

```
CREATE TABLE tablename (
  column_name data_type NOT NULL,
  CONSTRAINT pkname PRIMARY KEY (col),
  CONSTRAINT fkname FOREIGN KEY (col)
REFERENCES other_table(col_in_other_table),
  CONSTRAINT ucname UNIQUE (col),
  CONSTRAINT ckname CHECK (conditions)
);
```

Create Temporary Table

```
CREATE TEMPORARY TABLE
tablename (
  colname datatype
);
```

Drop Table

```
DROP TABLE tablename;
```

Alter Table

Add Column

```
ALTER TABLE tablename
ADD columnname datatype;
```

Drop Column

```
ALTER TABLE tablename
DROP COLUMN columnname;
```

Modify Column

```
ALTER TABLE tablename CHANGE
columnname newcolumnname newdatatype;
```

Rename Column

```
ALTER TABLE tablename CHANGE
COLUMN currentname TO newname;
```

Add Constraint

```
ALTER TABLE tablename ADD
CONSTRAINT constraintname
constrainttype (columns);
```

Drop Constraint

```
ALTER TABLE tablename DROP
constraint_type constraintname;
```

Rename Table

```
ALTER TABLE tablename
RENAME TO newtablename;
```

Window/Analytic Functions

```
function_name ( arguments ) OVER (
[query_partition_clause]
[ORDER BY order_by_clause
[windowing_clause] ] )
```

Example using RANK, showing the student details and their rank according to the fees_paid, grouped by gender:

```
SELECT
student_id, first_name, last_name, gender, fees_paid,
RANK() OVER (
  PARTITION BY gender ORDER BY fees_paid
) AS rank_val
FROM student;
```

Subqueries

Single Row

```
SELECT id, last_name, salary
FROM employee
WHERE salary = (
  SELECT MAX(salary)
  FROM employee
);
```

Multi Row

```
SELECT id, last_name, salary
FROM employee
WHERE salary IN (
  SELECT salary
  FROM employee
  WHERE last_name LIKE 'C%'
);
```

PostgreSQL Cheat Sheet

www.databasestar.com

SELECT Query

```
SELECT col1, col2
FROM table
JOIN table2 ON table1.col = table2.col
WHERE condition
GROUP BY column_name
HAVING condition
ORDER BY col1 ASC|DESC;
```

SELECT Keywords

DISTINCT: Removes duplicate results

```
SELECT DISTINCT product_name
FROM product;
```

BETWEEN: Matches a value between two other values (inclusive)

```
SELECT product_name
FROM product
WHERE price BETWEEN 50 AND 100;
```

IN: Matches to any of the values in a list

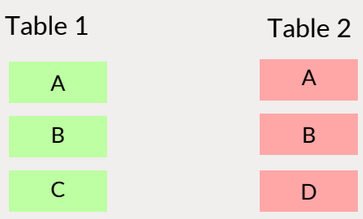
```
SELECT product_name
FROM product
WHERE category IN
('Electronics', 'Furniture');
```

LIKE: Performs wildcard matches using _ or %

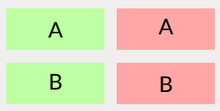
```
SELECT product_name
FROM product
WHERE product_name
LIKE '%Desk%";
```

Joins

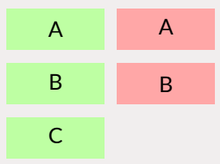
```
SELECT t1.*, t2.*
FROM t1
join_type t2 ON t1.col = t2.col;
```



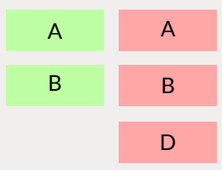
INNER JOIN: show all matching records in both tables.



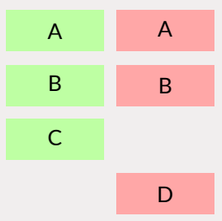
LEFT JOIN: show all records from left table, and any matching records from right table.



RIGHT JOIN: show all records from right table, and any matching records from left table.



FULL JOIN: show all records from both tables, whether there is a match or not.



CASE Statement

Simple Case

```
CASE name
  WHEN 'John' THEN 'Name John'
  WHEN 'Steve' THEN 'Name Steve'
  ELSE 'Unknown'
END
```

Searched Case

```
CASE
  WHEN name='John' THEN 'Name John'
  WHEN name='Steve' THEN 'Name Steve'
  ELSE 'Unknown'
END
```

Common Table Expression

```
WITH queryname AS (
  SELECT col1, col2
  FROM firsttable)
SELECT col1, col2..
FROM queryname...;
```

Modifying Data

Insert

```
INSERT INTO tablename
(col1, col2...)
VALUES (val1, val2);
```

Insert from a Table

```
INSERT INTO tablename
(col1, col2...)
SELECT col1, col2...
```

Insert Multiple Rows

```
INSERT INTO tablename
(col1, col2...) VALUES
(valA1, valB1),
(valA2, valB2),
(valA3, valB3);
```

Update

```
UPDATE tablename
SET col1 = val1
WHERE condition;
```

Update with a Join

```
UPDATE t
SET col1 = val1
FROM tablename t
INNER JOIN table x
ON t.id = x.tid
WHERE condition;
```

Delete

```
DELETE FROM tablename
WHERE condition;
```

Indexes

Create Index

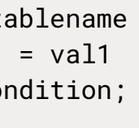
```
CREATE INDEX indexname
ON tablename (cols);
```

Drop Index

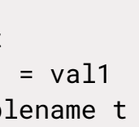
```
DROP INDEX indexname;
```

Set Operators

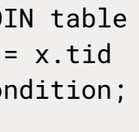
UNION: Shows unique rows from two result sets.



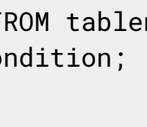
UNION ALL: Shows all rows from two result sets.



INTERSECT: Shows rows that exist in both result sets.



EXCEPT: Shows rows that exist in the first result set but not the second.



Aggregate Functions

- SUM: Finds a total of the numbers provided
- COUNT: Finds the number of records
- AVG: Finds the average of the numbers provided
- MIN: Finds the lowest of the numbers provided
- MAX: Finds the highest of the numbers provided

Common Functions

- LENGTH(string): Returns the length of the provided string
- POSITION(string IN substring): Returns the position of the substring within the specified string.
- CAST(expression AS datatype): Converts an expression into the specified data type.
- NOW: Returns the current date, including time.
- CEIL(input_val): Returns the smallest integer greater than the provided number.
- FLOOR(input_val): Returns the largest integer less than the provided number.
- ROUND(input_val, [round_to]): Rounds a number to a specified number of decimal places.
- TRUNC(input_value, num_decimals): Truncates a number to a number of decimals.
- REPLACE(whole_string, string_to_replace, replacement_string): Replaces one string inside the whole string with another string.
- SUBSTRING(string, [start_pos], [length]): Returns part of a value, based on a position and length.

Create Table

Create Table

```
CREATE TABLE tablename (
  column_name data_type
);
```

Create Table with Constraints

```
CREATE TABLE tablename (
  column_name data_type NOT NULL,
  CONSTRAINT pkname PRIMARY KEY (col),
  CONSTRAINT fkname FOREIGN KEY (col)
REFERENCES other_table(col_in_other_table),
  CONSTRAINT ucname UNIQUE (col),
  CONSTRAINT ckname CHECK (conditions)
);
```

Create Temporary Table

```
CREATE TEMP TABLE tablename (
  colname datatype
);
```

Drop Table

```
DROP TABLE tablename;
```

Alter Table

Add Column

```
ALTER TABLE tablename ADD COLUMN
columnname datatype;
```

Drop Column

```
ALTER TABLE tablename DROP COLUMN
columnname;
```

Modify Column

```
ALTER TABLE tablename ALTER COLUMN
columnname TYPE newdatatype;
```

Rename Column

```
ALTER TABLE tablename RENAME COLUMN
currentname TO newname;
```

Add Constraint

```
ALTER TABLE tablename ADD CONSTRAINT
constraintname constrainttype
(columns);
```

Drop Constraint

```
ALTER TABLE tablename DROP
constraint_type constraintname;
```

Rename Table

```
ALTER TABLE tablename
RENAME TO newtablename;
```

Window/Analytic Functions

```
function_name ( arguments ) OVER (
[query_partition_clause]
[ORDER BY order_by_clause
[windowing_clause] ] )
```

Example using RANK, showing the student details and their rank according to the fees_paid, grouped by gender:

```
SELECT
student_id, first_name, last_name, gender, fees_paid,
RANK() OVER (
  PARTITION BY gender ORDER BY fees_paid
) AS rank_val
FROM student;
```

Subqueries

Single Row

```
SELECT id, last_name, salary
FROM employee
WHERE salary = (
  SELECT MAX(salary)
  FROM employee
);
```

Multi Row

```
SELECT id, last_name, salary
FROM employee
WHERE salary IN (
  SELECT salary
  FROM employee
  WHERE last_name LIKE 'C%'
);
```